**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

Тема: **ДВУМЕРНЫЕ СТАТИЧЕСКИЕ МАССИВЫ. УКАЗАТЕЛИ**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2372 |  | Астафьева У.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение двумерные статические массивов: объявление массивов, ввод-вывод массивов; получение практических навыков работы с двумерными статическими массивами, указателями;

**Основные теоретические положения.**

Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Наглядно одномерный массив можно представить, как набор пронумерованных ячеек, в каждой из которых содержится определенное значение. Каждый элемент в этих массивах определяется значением индекса элемента.

Все массивы можно разделить на две группы: *одномерные* и *многомерные*. Многомерные массивы определяются аналогично одномерным массивам. Количество элементов по каждому измерению указывается отдельно в квадратных скобках. Для доступа к определенному элементу многомерного массива необходимо указать в квадратных скобках конкретные значения всех индексов этого элемента.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных. Так же, как и в одномерном массиве, элементы многомерных массивов располагаются друг за другом в непрерывном участке памяти.

Значения индексов элементов массивов всегда начинается с 0. Поэтому максимальное значение индекса элемента в массиве всегда на единицу меньше количества элементов в массиве.

Ввод и вывод массивов реализуются с помощью циклов.

Указатели – это обычные переменные, но они служат для хранения адресов памяти.

Указатели определяются в программе следующим образом:

<тип данных>\* <имя переменной>

Формально указатели представляют собой обычные целые значения типа int и занимают в памяти 4 байта не зависимо от базового типа указателя. Значения указателей при их выводе на экран представляются как целые значения в шестнадцатеричном формате.

Указатели поддерживают ряд операций: присваивание, получение адреса указателя, получение значения по указателю, некоторые арифметические операции и операции сравнения.

**Постановка задачи.**

Необходимо написать программу, которая:

1)    Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка *N* (6,8,10) случайными числами от 1 до N\*N согласно схемам, приведенным на рисунках (спираль, змейка). Пользователь должен видеть процесс заполнения квадратичной матрицы.

2)    Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами.

3)    Используя арифметику указателей, сортирует элементы любой сортировкой.

4)    Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число.

**Выполнение работы.**

Код программы представлен в приложении А.

1. При запуске программы пользователю выводится меню доступных команд и ожидается ввод команды с клавиатуры.

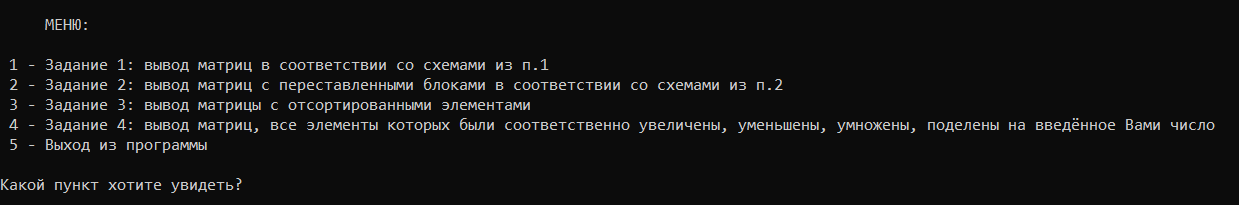


Рисунок 1. – Запуск программы, меню

1. Следующий шаг зависит от введенной команды, если пользователь ввёл:

2.1. “1”, то массив выводится в соответствии со схемами: спиралью и змейкой.

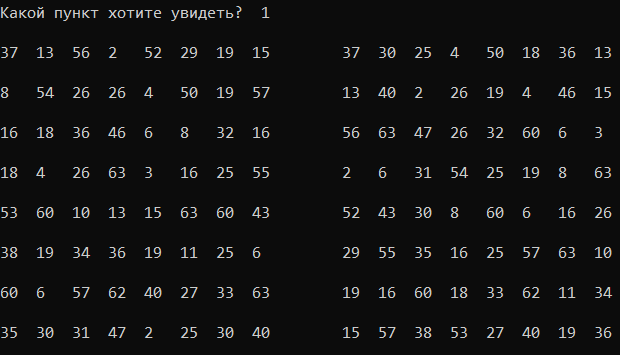


Рисунок 2. Задача 1 – Вывод матрицы спиралью и змейкой

2.2. “2”, то сначала выводится исходная матрица (для проверки перестановки блоков), а потом выводятся матрицы с переставленными блоками в соответствии со схемами.

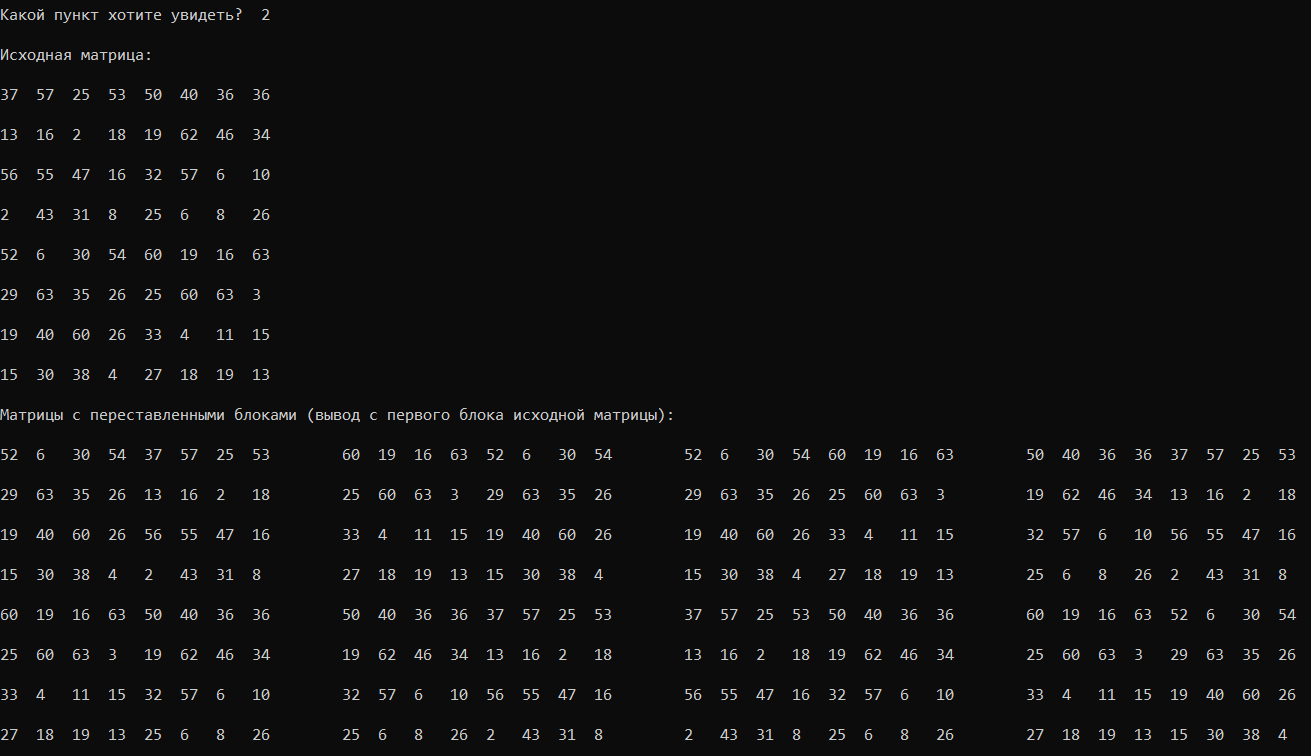


Рисунок 3. Задача 2 – Новые матрицы, полученные путём перестановки блоков исходной матрицы в соответствии со схемами

* 1. “3”, то выводится матрица с отсортированными по возрастанию элементами.

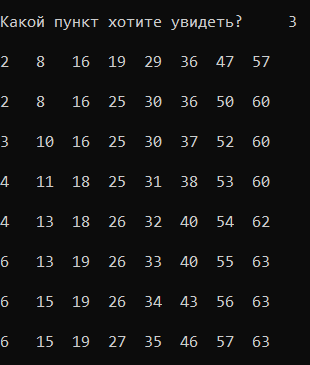


Рисунок 4. Задача 3 – Матрица с отсортированными элементами

* 1. “4”, то сначала пользователю предлагается ввести число, на которое уменьшатся, увеличатся, умножатся и поделятся все элементы матрицы.

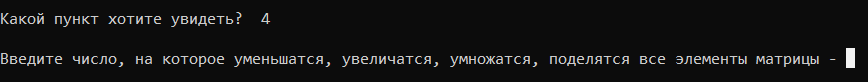


Рисунок 5. Задача 4 – ввод числа

А затем выводятся матрицы с соответственно преобразованными элементами.

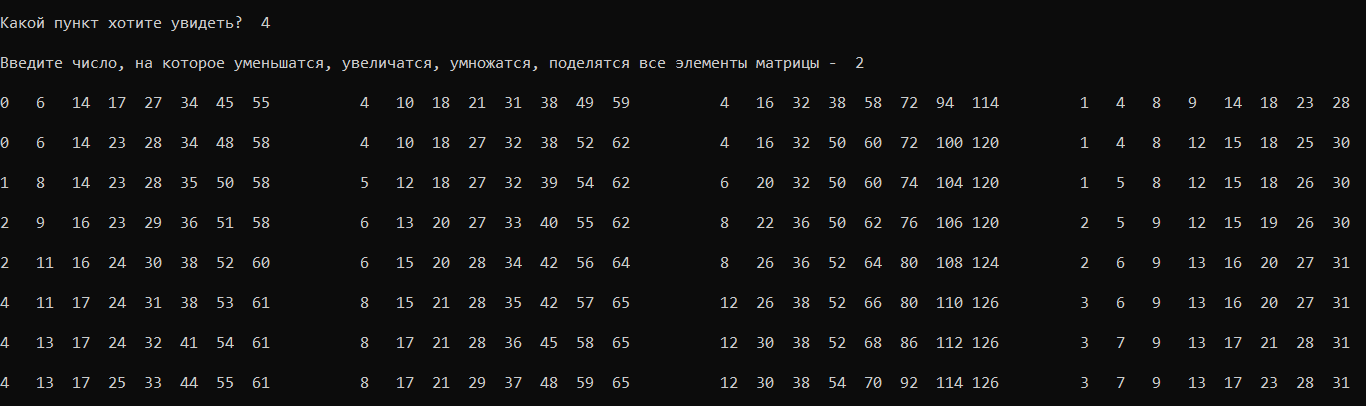


Рисунок 6. Задача 4 – Матрицы с элементами, уменьшенными, увеличенными, умноженными, поделенными на введённое пользователем число

* 1. “0”, от “6” до любого другого числа, то пользователю предлагается ввести другое число, так как команды с таким номером нет.

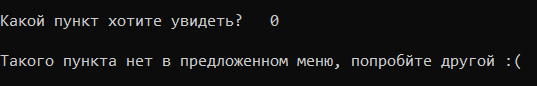


Рисунок 7 – Вывод пункта, не предложенного в меню

* 1. “5”, то происходит выход из программы.

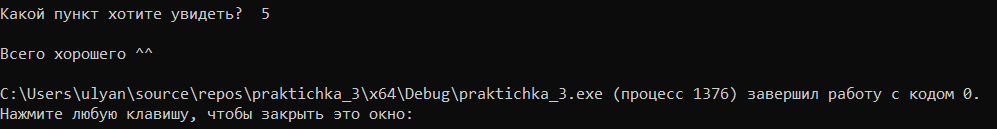
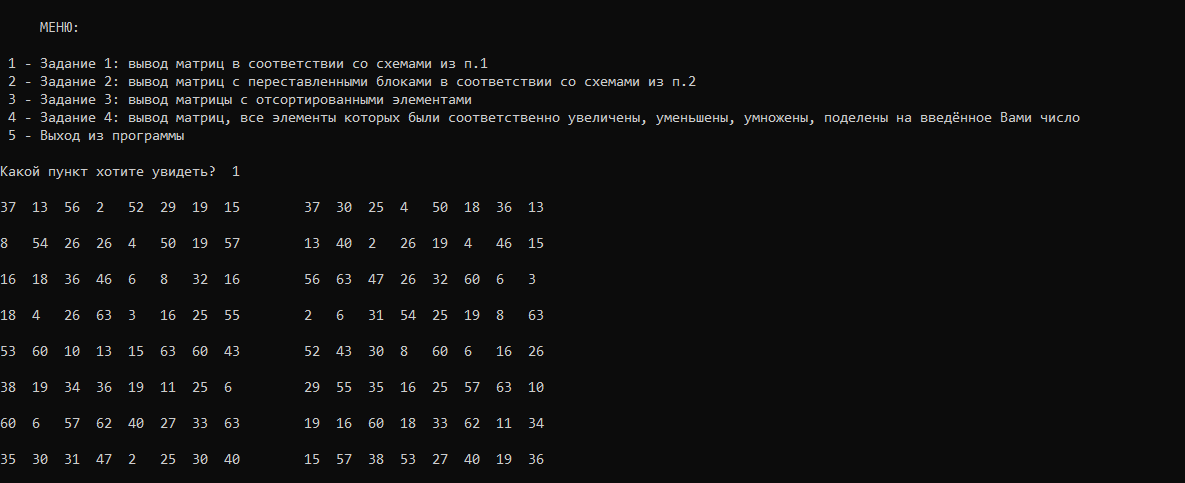
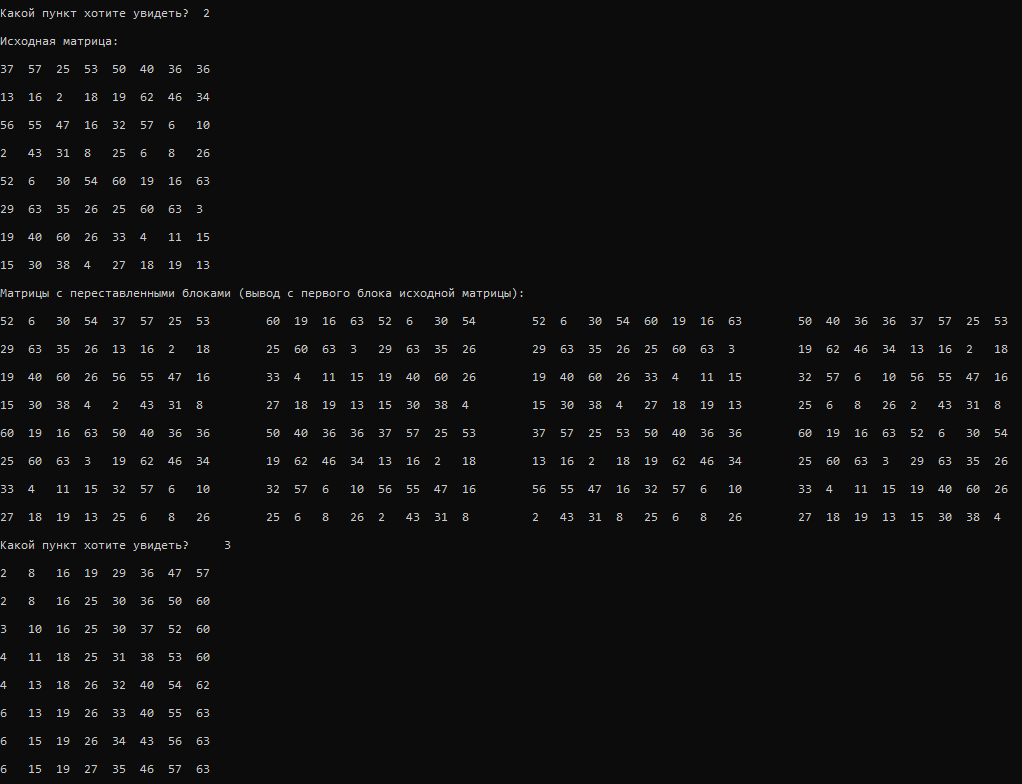
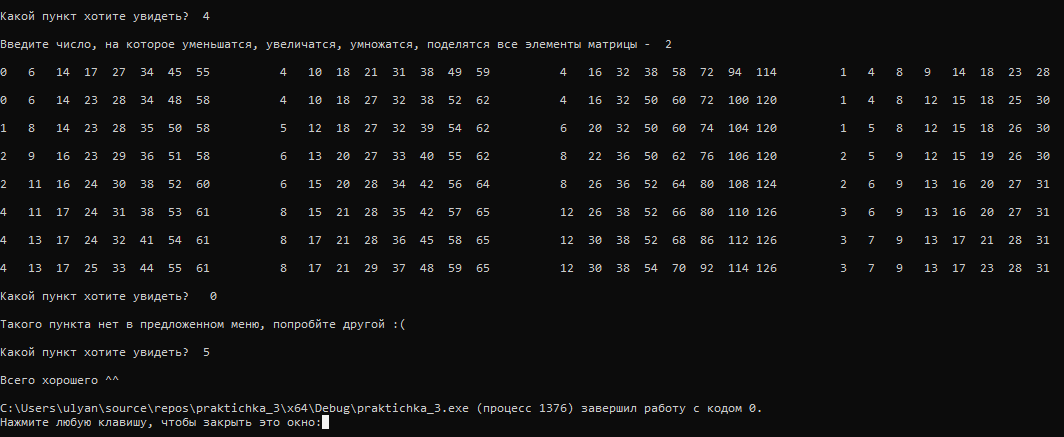


Рисунок 8 – Выход из программы

Общий результат кода с примерами:







**Выводы.**

В ходе работы было изучено: объявление двумерных статических массивов, ввод-вывод массивов, работа с указателями, связь массивом и указателей.

Приложение А

рабочий код

#include <iostream>

#include <Windows.h>

#include <algorithm>

using namespace std;

void matrix(int arr[], int n, int& elemMass, int coordY)

{

int coord = coordY;

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0; i < (n \* 4); i += 4)

{

for (int j = coord; j < (n \* 2) + coord; j += 2)

{

destCoord.X = i;

destCoord.Y = j;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(70);

}

}

}

void spiral(int arr[], int n, int koffX, int koffY, int& elemMass, int coordY)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0 + koffX; i < (n \* 4); i += 4)

{

destCoord.X = i;

destCoord.Y = coordY + 0 + koffY;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(100);

}

for (int i = coordY + 2 + koffY; i < (n \* 2) + coordY; i += 2)

{

destCoord.X = n \* 4 - 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(100);

}

for (int i = (n \* 4 - 4 \* 2); i > (-1 + koffX); i -= 4)

{

destCoord.X = i;

destCoord.Y = coordY + n \* 2 - 2;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(100);

}

for (int i = (coordY + n \* 2 - 2 \* 2); i > (1 + koffY) + coordY; i -= 2)

{

destCoord.X = 0 + koffX;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(90);

}

while (n > (n / 2))

return spiral(arr, n - 1, koffX + 4, koffY + 2, elemMass, coordY);

}

void snake(int arr[], int n, int koffX, int koffY, int& t, int coordY)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0 + coordY; i < (n \* 2) + coordY; i += 2)

{

destCoord.X = (n \* 4 + 6) + koffX \* 2;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + t);

t += 1;

cout.flush();

Sleep(100);

}

for (int i = (n \* 2) - 2 + coordY; i > (-1) + coordY; i -= 2)

{

destCoord.X = ((n \* 4 + 6) + 4) + koffX \* 2;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + t);

t += 1;

cout.flush();

Sleep(90);

}

while (t < (n \* n))

return snake(arr, n, koffX + 4, koffY + 2, t, coordY);

}

void matrix4(int arr[], int n, int& elemMass, int coordY, int coordX)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

elemMass = 0;

for (int i = 0 + coordX; i < (n \* 4) + coordX; i += 4)

{

for (int j = coordY; j < (n \* 2) + coordY; j += 2)

{

destCoord.X = i;

destCoord.Y = j;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(90);

}

}

}

void block(int arr[], int n, int& elemMass, int coordY, int coordX)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0 + coordX; i < ((n / 2) \* 4) + coordX; i += 4)

{

for (int j = coordY; j < ((n / 2) \* 2) + coordY; j += 2)

{

destCoord.X = i;

destCoord.Y = j;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(90);

}

elemMass += (n/2);

}

}

void sort(int arr[], int n)

{

int\* p = arr;

for (int i = 1; i < (n \* n); i++)

for (int j = i; \*(p + j - 1) > \*(p + j); j--)

swap(\*(p + j - 1), \*(p + j));

}

void Blinking(int off = 99, bool mood = 0)

{

HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO cci{};

cci.dwSize = off;

cci.bVisible = mood;

SetConsoleCursorInfo(hStdout, &cci);

}

int main()

{

setlocale(0, "");

srand(time(NULL));

const int n = 8;

int arr[n \* n]{};

int koffX = 0, koffY = 0;

int elemMass = 0, t = 0, point, coordY = 0, coordX = 0;

bool flag = true;

for (int v = 0; v < (n \* n); v++)

arr[v] = rand() % (n \* n) + 1;

cout << "\n МЕНЮ:\n\n 1 - Задание 1: вывод матриц в соответствии со схемами из п.1\n" \

" 2 - Задание 2: вывод матриц с переставленными блоками в соответствии со схемами из п.2\n" \

" 3 - Задание 3: вывод матрицы с отсортированными элементами\n" \

" 4 - Задание 4: вывод матриц, все элементы которых были соответственно увеличены, уменьшены, умножены, поделены на введённое Вами число \n" \

" 5 - Выход из программы\n\n";

while (flag)

{

Blinking(99, 1);

cout << "Какой пункт хотите увидеть? ";

cin >> point;

Blinking();

switch (point)

{

case 1:

{

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

elemMass = 0;

t = 0;

spiral(arr, n, koffX, koffY, elemMass, coordY);

snake(arr, n, koffX, koffY, t, coordY);

for (int i = 0; i < (n\*2); i++)

cout << "\n";

break;

}

case 2:

{

cout << "\nИсходная матрица: \n";

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

elemMass = 0;

matrix(arr, n, elemMass, coordY);

cout << "\n\nМатрицы с переставленными блоками (вывод с первого блока исходной матрицы): \n";

CONSOLE\_SCREEN\_BUFFER\_INFO fi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &fi);

coordY = fi.dwCursorPosition.Y + 1;

coordX = (n / 2) \* 4;

elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

coordY += (n / 2) \* 2;

elemMass = (n\*n)/2;

block(arr, n, elemMass, coordY, coordX);

coordX = 0;

elemMass = (n\*n)/2 + 4;

block(arr, n, elemMass, coordY, coordX);

elemMass = n/2;

coordY -= (n / 2) \* 2;

block(arr, n, elemMass, coordY, coordX);

coordX = (n \* 4) + 6 + (n / 2) \* 4;

coordY += (n / 2) \* 2;

elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

coordX -= (n / 2) \* 4;

elemMass = (n \* n) / 2;

block(arr, n, elemMass, coordY, coordX);

elemMass = (n \* n) / 2 + 4;

coordY -= (n / 2) \* 2;

block(arr, n, elemMass, coordY, coordX);

elemMass = n / 2;

coordX += (n / 2) \* 4;

block(arr, n, elemMass, coordY, coordX);

coordX = (n \* 4) \* 2 + 6 \* 2;

coordY += (n / 2) \* 2;

elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

elemMass = (n \* n) / 2;

coordX += (n / 2) \* 4;

block(arr, n, elemMass, coordY, coordX);

coordY -= (n / 2) \* 2;

elemMass = (n \* n) / 2 + 4;

block(arr, n, elemMass, coordY, coordX);

coordX -= (n / 2) \* 4;

elemMass = n / 2;

block(arr, n, elemMass, coordY, coordX);

coordX = (n \* 4) \* 3 + 6 \* 3 + (n / 2) \* 4;

elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

coordX -= (n / 2) \* 4;

elemMass = (n \* n) / 2;

block(arr, n, elemMass, coordY, coordX);

coordY += (n / 2) \* 2;

elemMass = (n \* n) / 2 + 4;

block(arr, n, elemMass, coordY, coordX);

coordX += (n / 2) \* 4;

elemMass = n / 2;

block(arr, n, elemMass, coordY, coordX);

cout << "\n\n";

break;

}

case 3:

{

elemMass = 0;

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

sort(arr, n);

matrix(arr, n, elemMass, coordY);

cout << "\n\n";

break;

}

case 4:

{

int num, coordX = 0;

int copyArr[n \* n]{};

elemMass = 0;

cout << "\nВведите число, на которое уменьшатся, увеличатся, умножатся, поделятся все элементы матрицы - ";

cin >> num;

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] - num;

matrix4(copyArr, n, elemMass, coordY, coordX);

coordX = n \* 4 + 8;

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] + num;

matrix4(copyArr, n, elemMass, coordY, coordX);

coordX += n \* 4 + 8;

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] \* num;

matrix4(copyArr, n, elemMass, coordY, coordX);

coordX += n \* 4 + 8;

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] / num;

matrix4(copyArr, n, elemMass, coordY, coordX);

cout << "\n\n";

break;

}

case 5:

cout << "\nВсего хорошего ^^\n";

flag = false;

break;

default:

cout << "\nТакого пункта нет в предложенном меню, попробйте другой :(\n\n";

}

}

return 0;

}